Efficient Routing in Packet-Radio Networks Using Link-State Information

J.J. Garcia-Luna-Aceves
Computer Engineering Department
University of California
Santa Cruz, California 95064
jj@cse.ucsc.edu

Marcelo Spohn
Rooftop Communications
Mountain View, California 94041
marcelo@rooftop.com

Abstract- **We present the source-tree adaptive routing (STAR) protocol, which we show through simulation experiments to be far more efficient than the Dynamic Source Routing (DSR) protocol, which has been shown to be one of the best performing on-demand routing protocols. A router in STAR communicates to its neighbors the parameters of its source routing tree, which consists of each link that the router needs to reach every destination. To conserve transmission bandwidth and energy, a router transmits changes to its source routing tree only when the router detects new destinations, the possibility of looping, or the possibility of node failures or network partitions.**

I. Introduction

Multi-hop packet-radio networks, or ad-hoc networks, consist of mobile hosts interconnected by routers that can also move. The deployment of such routers is ad-hoc and the topology of the network is very dynamic, because of host and router mobility, signal loss and interference, and power outages. In addition, the channel bandwidth available in ad-hoc networks is relatively limited compared to wired networks, and untethered routers may need to operate with battery-life constraints.

Routing algorithms for ad-hoc networks can be categorized according to the way in which routers obtain routing information, and according to the type of information they use to compute preferred paths. In terms of the way in which routers obtain information, routing protocols have been classified as table-driven and on-demand. In terms of the type of information used by routing protocols, routing protocols can be classified into link-state protocols and distance-vector protocols. Routers running a link-state protocol use topology information to make routing decisions; routers running a distance-vector protocol use distances and, in some cases, path information, to destinations to make routing decisions.

In an on-demand routing protocol, routers maintain path information for only those destinations that they need to contact as a source or relay of information. The basic approach consists of allowing a router that does not know how to reach a destination to send a flood-search message to obtain the path information it needs. The first routing protocol of this type was proposed to establish virtual circuits in the MSE network [9], and there are several more recent examples of this approach (e.g., AODV [13], DSR [7], TORA [11]). The Dynamic Source Routing (DSR) protocol has been shown to outperform many other on-demand routing protocols [2]. All of the on-demand routing protocols reported to date are based on distances to destinations, and there have been no on-demand link-state proposals to date.

In a table-driven algorithm, each router maintains path information for each known destination in the network and updates its routing-table entries as needed. Examples of table-driven algorithms based on distance vectors are the routing protocol of the DARPA packet-radio network [8], DSDV [12], WRP [15], and WIRP [4]. Prior table-driven approaches to link-state routing in packet-radio networks are based on topology broadcast. However, disseminating complete link-state information to all routers incurs excessive communication overhead in an ad-hoc network because of the dynamics of the network and the small bandwidth available. Accordingly, all link-state routing approaches for packet-radio networks have been based on hierarchical routing schemes [14], [16].

To date, the debate on whether a table-driven or an on-demand routing approach is best for wireless networks has assumed that table-driven routing necessarily has to provide optimum (e.g., shortest-path) routing, when in fact on-demand routing protocols cannot ensure optimum paths. In this paper, we introduce the source-tree adaptive routing (STAR) protocol, which is the first example of a table-driven routing protocol that is more efficient than any on-demand routing protocol by exploiting link-state information and allowing paths taken to destinations to deviate from the optimum in order to save bandwidth. Furthermore, STAR can be used with distributed hierarchical routing schemes proposed in the past for both distance-vector or link-state routing.

Section II describes STAR, and Section III compares STAR's performance against the performance of DSR, which has been shown to incur the least overhead among several on-demand routing protocols [2].

II. STAR Description

A. Overview

In STAR, each router reports to its neighbors the characteristics of every link it uses to reach a destination. The set of links used by a router in its preferred path to destinations is called the *source tree* of the router. A router knows its adjacent links and the source trees reported by its neighbors; the aggregation of a router's adjacent links and the source trees reported by its neighbors constitute a partial *topology graph*. The links in the source tree and topology graph must be adjacent links or links reported by at least one neighbor. The router uses the topology graph to generate its own source tree. Each router derives a routing table specifying the successor to each destination by running a local *route-selection algorithm* on its source tree. Although any type of local route-selection algorithm can be used in STAR, we describe STAR assuming that Dijkstra's shortest-path first (SPF) algorithm is used at each router to compute preferred paths.

A router running STAR sends updates on its source tree to its neighbors only when it loses all paths to one ore more destinations, when it detects a new destination, or when it determines that local changes to its source tree can potentially create long term routing loops. Because each router communicates its source tree to its neighbors, the deletion of a link no longer used to reach a destination is implicit with the addition of the new link used to reach the destination and need not be sent explicitly as an update; a router makes explicit reference to a failed link only when the deletion of a link causes the router to have no paths to

This work was supported in part at UCSC by the Defense Advanced Research Projects Agency (DARPA) under Grant F30602-97-2-0338.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
1999		**00-00-1999 to 00-00-1999**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Efficient Routing in Packet-Radio Networks Using Link-State Information	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California,Computer Engineering Department,Santa Cruz,CA,95064**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**5**	

one or more destinations, in which case the router cannot provide new links to make the deletion of the failed link implicit.

The basic update unit used in STAR to communicate changes to source trees is the link-state update (LSU). An LSU for a link (u, v) in an update message is a tuple (u, v, l, sn) reporting the characteristics of the link, where l represents the cost of the link and sn is the sequence number assigned to the LSU; an update message contains one or more LSUs. For a link between router u and router or destination v, router u is called the *head node* of the link in the direction from u to v. The head node of a link is the only router that can report changes in the parameters of that link.

LSUs are validated using sequence numbers, a router receiving an LSU accepts the LSU as valid if the received LSU has a larger sequence number than the sequence number of the LSU stored from the same source, or if there is no entry for the link in the topology graph and the LSU is not reporting an infinite cost. Link-state information for failed links are the only LSUs erased from the topology graph due to aging (which is in the order of an hour after having processed the LSU). LSUs for operational links are erased from the topology graph when the links are erased from the source tree of all the neighbors.

We note that, because LSUs for operational links never age out, there is no need for the head node of a link to send periodic LSUs to update the sequence number of the link. This is very important, because it means that STAR does not need periodic update messages to validate link-state information like OSPF [10] and every single routing protocol based on sequence numbers or time stamps does! To simplify our description, the specification in the rest of this paper describes STAR as if unbounded counters were available to keep track of sequence numbers.

An underlying protocol, which we call the neighbor protocol, assures that a router detects within a finite time the existence of a new neighbor and the loss of connectivity with a neighbor, and provides the reliable transmission of update messages generated by STAR to the neighbors of the router. All messages, changes in the cost of a link, link failures, and new-neighbor notifications are processed one at a time within a finite time and in the order in which they are detected. Routers are assumed to operate correctly, and information is assumed to be stored without errors. The cost of a failed link is considered to be infinity.

B. Exchanging Update Messages

We can distinguish between two main approaches to updating routing information in the routing protocols that have been designed for wireless networks: the *optimum routing approach* (ORA) and the *least-overhead routing approach* (LORA). With ORA, the routing protocol attempts to update routing tables as quickly as possible to provide paths that are optimum with respect to a defined metric. In contrast, with LORA, the routing protocol attempts to provide viable paths according to a given performance metric, which need not be optimum, to incur the least amount of control traffic.

On-demand routing protocols such as DSR follow LORA. Interestingly, all the table-driven routing protocols reported to date for ad-hoc networks adhere to ORA, and admittedly have been adaptations of routing protocols developed for wired networks; STAR is the first table-driven routing protocol that implements LORA.

In an on-demand routing protocol, a router can keep using a path found as long as the path leads to the destination, even if the path does not have optimum cost. A similar approach is used in STAR, because each router has a complete path to every destination as part of its source tree. A router i running STAR should send update messages according to the following three rules, which inform routers of unreachable destinations, new destinations, and update topology information to prevent permanent routing loops. Router i implements these rules by comparing its source tree against the source trees it has received from its neighbors.

LORA-1: Router i finds a new destination, or any of its neighbors reports a new destination.

Whenever a router hears from a new neighbor that is also a new destination, it sends an update message that includes the new LSUs in its source tree. Obviously, when a router is first initialized or after a reboot, the router itself is a new destination and should send an update message to its neighbors. Link-level support should be used for the router to know its neighbors within a short time, and then report its links to those neighbors with LSUs sent in an update message. Else, a simple way to implement an initialization action consists of requiring the router to listen for some time for neighbor traffic, so that it can detect the existence of links to neighbors.

LORA-2: At least one destination becomes unreachable to router i or any of its neighbors.

When a router processes an input event (e.g., a link fails, an update message is received) that causes *all* its paths through all its neighbors to one or more destination to be severed, the router sends an update message that includes an LSU specifying an infinite cost for the link connecting to the head of each subtree of the source tree that becomes unreachable. The update message does not have to include an LSU for each node in an unreachable subtree, because a neighbor receiving the update message has the sending node's source tree and can therefore infer that all nodes below the root of the subtree are also unreachable, unless LSUs are sent for new links used to reach some of the nodes in the subtree.

LORA-3: This rule has three parts:

1. A path implied in the source tree of router i leads to a loop.
2. The new successor chosen to a given destination has an address larger than the address of router i.
3. The reported distance from the new chosen successor n to a destination j is longer than the reported distance from the previous successor to the same destination. However, if the link $(i,\ j)$ fails and n is a neighbor of j, no update message is needed regarding j or any destination whose path from i involves j.

Each time a router processes an update message from a neighbor, it updates that neighbor's source tree and traverses that tree to determine for which destinations its neighbor uses the router as a relay in its preferred paths. The router then determines if it is using the same neighbor as a relay for any of the same destinations. A routing loop is detected if the router and neighbor use each other as relay to any destination, in which case the loop must be broken and the router must send an update message with the corresponding changes.

To explain the need for the second part of LORA-3, we observe that, in any routing loop among routers with unique addresses, one of the routers must have the smallest address in the loop; therefore, if a router is forced to send an update message when it chooses a successor whose address is larger than its own, then it is not possible for all routers in a routing loop to remain quiet after choosing one another, because at least one of them is forced to send an update message, which causes the loop to break when routers update their source trees.

The last part of LORA-3 is needed when link costs can assume different values in different directions, in which case the second part of LORA-3 may not suffice to break loops because the node with the smallest address in the loop may not have to change successors when the loop is formed. The following example illustrates this scenario. Consider the six-node wireless network shown in Figure 1 and assume

that the third part of LORA-3 is not in effect at the routers running STAR. In this example, nodes are given identifiers that are lexicographically ordered, i.e., a is the smallest identifier and f is the largest identifier in the graph. All links and nodes are assumed to have the same propagation delays, and all the links but link (b, c) have unit cost. Figures 1(b) through 1(d) show the source trees according to STAR at the routers indicated with filled circles for the network topology depicted in Figure 1(a). Arrowheads on solid lines indicate the direction of the links stored in the router's source tree. Figure 1(e) shows c's new source tree after processing the failure of link (c, d); we note that c does not generate an update message, because $c > b$ by assumption. Suppose link (b, e) fails immediately after the failure of (c, d), node b computes its new source tree shown in Figure 1(f) without reporting changes to it because a is its new successor to destinations d, e, and f, and $a < b$. A permanent loop forms among nodes a, b, and c.

Figure 2 depicts the sequence of events triggered by the execution of the third part of LORA-3 in the same example introduced in Figure 1, after the failures of links (c, d) and (b, e). The figure shows the LSUs generated by the node with filled circle transmitted in an update message to the neighbors, and shows such LSUs in parentheses. The third element in an LSU corresponds to the cost of the link. Unlike in the previous example, node c transmits an update message after processing the failure of link (c, d) because of the third part of LORA-3; the distance from the new successor b to d and f is longer than from the previous successor d. When link (b, e) fails, node b realizes that the destinations d, e, and f are unreachable and generates an update message reporting the failure of the link connecting to the head of the subtree of the source tree that becomes unreachable. The update message from b triggers the update messages that allow nodes a, b, and c to realize that there are no paths to d, e, and f. A similar sequence of events takes place at the other side of the network partition.

The example shown in Figure 3 illustrates the scenario in which a router that chooses a new successor to a destination with a larger distance to it does not need to send an update message. For this example, the source trees of nodes a, b, and c are depicted in Figures 2(c), 1(c), and 2(b), respectively. Figure 3(b) shows the new source tree of node c after the failure of link (c, b). In this case, c does not need to send an update message because the parent node of the subtree headed by b is a neighbor of c and therefore no permanent loop can be formed.

To ensure that the above rules work with incremental updates specifying only changes to a source tree, a router must remember the source tree that was last notified to its neighbors. If any of LORA-1 to LORA-3 are satisfied, the router must do one of two things:

- If the new source tree includes new neighbors than those present in the source tree that was last updated, then the router must send its entire source tree in its update, so that new neighbors learn about all the destinations the router knows.
- If the two source trees imply the same neighbors, the router sends only the updates needed to obtain the new tree from the old one.

To ensure that STAR stops sending update messages, a simple rule can be used to determine which router must stop using its neighbor as a relay, such a rule can be, for example, "the router with the smaller address must change its path."

The above rules are sufficient to ensure that every router obtains loopless paths to all known destinations, without the routers having to send updates periodically. In addition to the ability for a router to detect loops in STAR, the two key features that enable STAR to adopt LORA are: (a) validating LSUs without the need of periodic updates, and (b) the ability to either listen to neighbors' packets or use a neighbor protocol at the link layer to determine who the neighbors of a router are.

If ORA is to be supported in STAR, the only rule needed for sending update messages consists of a router sending an update message every time its source tree changes.

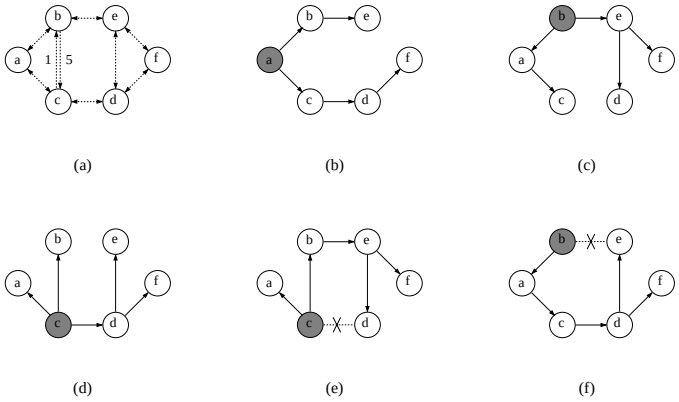


Fig. 1. Routers running STAR without the third part of LORA-3 in effect.

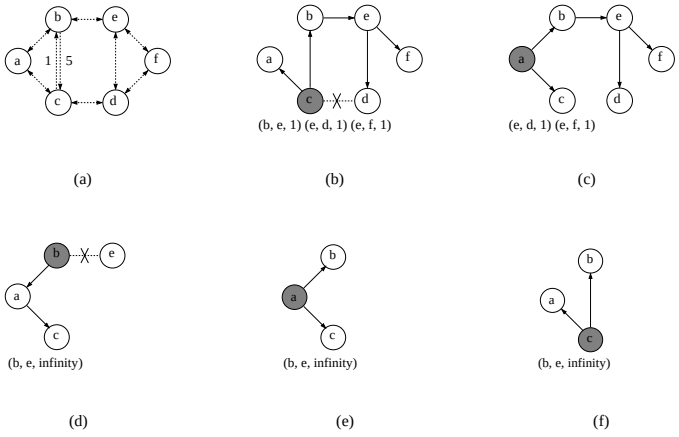


Fig. 2. Routers running STAR with the third part of LORA-3 in effect.

Fig. 3. The third part of LORA-3 not always triggers the generation of an update message: (a) network topology, and (b) the new source tree of node c after processing the failure of link (c, b).

III. Performance Evaluation

STAR has the same communication, storage, and time complexity than ALP [5] and efficient table-driven distance-vector routing protocols proposed to date (e.g., WRP [15]). However, worst-case performance is not truly indicative of STAR's performance.

We compare STAR with DSR, because DSR has been shown to be one of the best-performing on-demand routing protocols [2].

The protocol stack implementation in our simulator runs the very same code used in a real embedded wireless router and IP (Internet Protocol) is used as the network protocol.

The link layer implements a medium access control (MAC) protocol similar to the IEEE 802.11 [6] standard and the physical layer is based

on a direct sequence spread spectrum radio with a link bandwidth of 1 Mbit/sec. The neighbor protocol is configured to report loss of connectivity to a neighbor if the probation of the link fails in a period of about 10 seconds.

The simulation experiments use 20 nodes forming an ad-hoc network, moving over a flat space (5000m x 7000m), and initially randomly distributed at a density of one node per square kilometer.

The simulation study was conducted in the C++ Protocol Toolkit (CPT) simulator environment.

The simulation experiments use the same methodology used recently to evaluate DSR and other on-demand routing protocols [2]. To run DSR in our simulation environment, we ported the ns2 code available from [17] into the CPT simulator. There are only two differences in our DSR implementation with respect to that used in [2]: (1) in the embedded wireless routers and simulated protocol stack we used there is no access to the MAC layer and cannot reschedule packets already scheduled for transmission over a link (however, this is the case for all the protocols we simulate), and (2) routers cannot operate their network interfaces in *promiscuous mode* because the MAC protocol operates over multiple channels and a router does not know on which channels its neighbors are transmitting, unless the packets are meant for the router. Both STAR and DSR can buffer 20 packets that are awaiting discovery of a route through the network.

The overall goal of the simulation experiments was to measure the ability of the routing protocols to react to changes in the network topology while delivering data packets to their destinations. To measure this ability we applied to the simulated network three different communication patterns corresponding to 8, 14, and 20 data flows. The total workload in the three scenarios was the same and consisted of 32 data packets/sec, in the scenario with 8 flows each continuous bit rate (CBR) source generated 4 packets/sec, in the scenario with 20 sources each CBR source generated 1.6 packets/sec, and in the scenario with 14 flows there were 7 flows from distinct CBR sources to the same destination D generating an aggregate of 28 packets/sec and 7 flows having D as the CBR source and the other 7 sources of data as destinations. In all the scenarios the number of unique destinations was 8 and the packet size was 64 bytes. The data flows were started at times uniformly distributed between 20 and 120 seconds (we chose to start the flows after 20 seconds of simulated time to give some time to the Link Layer for determining the set of nodes that are neighbors of the routers).

The protocol evaluations are based on the simulation of 20 wireless nodes in continuous motion for 900 and 1800 seconds of simulated time.

Tables I and II summarize the behavior of STAR and DSR according to the simulated time. The tables show the total number of update packets transmitted by the nodes and the total number of data packets delivered to the applications for the three simulated workloads. The total number of update packets transmitted by routers running STAR varies with the number of changes in link connectivity while DSR generates control packets based on both variation of changes in connectivity and the type of workload inserted in the network. Routers running STAR generated less update packets than DSR in all simulated scenarios, the difference increased significantly when data traffic was inserted in the network for 1800 seconds: routers running DSR sent from 100% to 600% more update packets than STAR when nodes were moving during 1800 seconds of simulated time, and from 35% to 400% more update packets when nodes were moving during 900 seconds. Both STAR and DSR were able to deliver about the same number of data packets to the applications in the simulated scenarios with 8 and 14 flows. When we increased the number of sources of data from 8 to 20 nodes, while inserting the same number of data packets in the network (32 packets/sec), we observed that STAR was able to deliver as much as three times more data packets than DSR during 1800 seconds of simulated time, and almost twice the amount of data packets delivered by DSR during 900 seconds of simulated time.

Number of Flows	Update Pkts Sent		Data Pkts Delivered		Data Pkts Generated
	STAR	DSR	STAR	DSR	
8	585	791	14898	14740	24100
14	560	1466	15206	15367	25917
20	575	3122	13922	6830	23718

TABLE I

AVERAGE PERFORMANCE OF STAR AND DSR FOR NODES MOVING DURING 900 SECONDS OF SIMULATED TIME, TOTAL CHANGES IN LINK CONNECTIVITY WAS 1460.

Number of Flows	Update Pkts Sent		Data Pkts Delivered		Data Pkts Generated
	STAR	DSR	STAR	DSR	
8	946	1963	33159	32650	52900
14	911	2648	31992	32390	54716
20	934	6605	29978	10175	52518

TABLE II

AVERAGE PERFORMANCE OF STAR AND DSR FOR NODES MOVING DURING 1800 SECONDS OF SIMULATED TIME, TOTAL CHANGES IN LINK CONNECTIVITY WAS 2787.

The MAC layer discards all packets scheduled for transmission to a neighbor when the link to the neighbor fails, which contributes to the high loss of data packets seen by nodes. In DSR, each packet header carries the complete ordered list of routers through which the packet must pass and may be updated by nodes along the path towards the destination. The low throughput achieved by DSR for the case of 20 sources of data is due to the poor choice of source routes the routers make, leading to a significant increase in the number of ROUTE ERROR packets generated. Data packets are also discarded due to lack of routes to the destinations because the network may become temporarily partitioned or because the routing tables have not converged in the highly dynamic topology we simulate.

Figures 4(a) through 4(c) show the cumulative distribution of packet delay experienced by data packets during 1800 seconds of simulated time, for a workload of 8, 14, and 20 flows respectively. The higher delay introduced by DSR when relaying data packets is not directly

Number of Flows	Protocol	Number of Hops					
		1	2	3	4	5	6
8	STAR	92.0	7.4	0.2		0.4	
	DSR	64.9	31.2	2.6	1.3		
14	STAR	82.0	16.0	1.7		0.3	
	DSR	64.1	26.9	4.0	4.5	0.5	
20	STAR	92.6	5.1	2.0	0.3		
	DSR	61.9	32.4	5.1	0.3		0.3

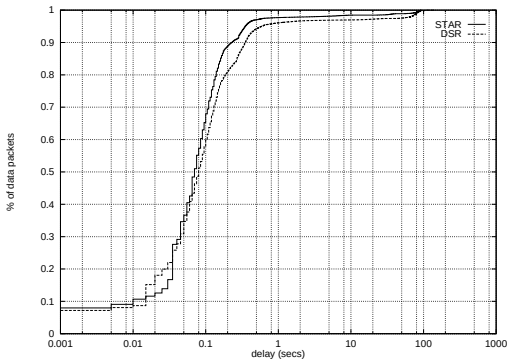
TABLE III

DISTRIBUTION OF DATA PACKETS DELIVERED ACCORDING TO THE NUMBER OF HOPS TRAVERSED (900 SECONDS OF SIMULATED TIME).

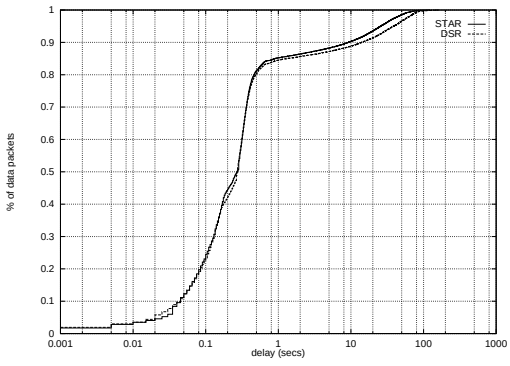
Number of Flows	Protocol	Number of Hops					
		1	2	3	4	5	6
8	STAR	92.8	6.7	0.3		0.2	
	DSR	64.3	29.2	5.9	0.6		
14	STAR	86.0	11.4	2.2	0.2	0.2	
	DSR	71.4	21.8	3.0	3.6	0.2	
20	STAR	91.1	6.9	1.4	0.5	0.1	
	DSR	67.5	28.7	3.5	0.2		0.1

TABLE IV

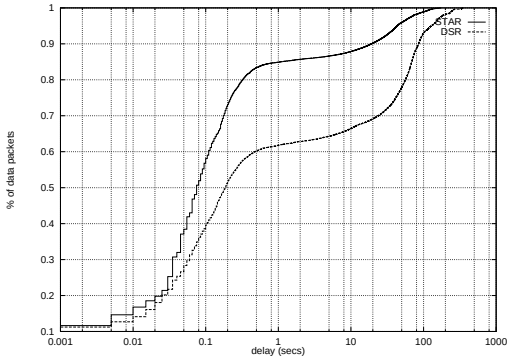
DISTRIBUTION OF DATA PACKETS DELIVERED ACCORDING TO THE NUMBER OF HOPS TRAVERSED (1800 SECONDS OF SIMULATED TIME).



(a)



(b)



(c)

Fig. 4. Cumulative distribution of packet delay experienced by data packets during 1800 seconds of simulated time for a workload of (a) 8 flows, (b) 14 flows, and (c) 20 flows.

related with the number of hops traversed by the packets (as shown in Tables III and IV) but with the poor choice of source routes when the number of flows increase from 8 to 20.

In all the simulation scenarios the number of destinations was set to just 40% of the number of nodes in the network in order to be fair with DSR. For the cases in which all the nodes in the network receive data, STAR would introduce no extra overhead while DSR could be severely penalized. It is also important to note the low ratio of update messages generated by STAR compared to the number of changes in link connectivity (Tables I and II).

We note that in cases where routers fail or the network becomes partitioned for extended time periods the bandwidth consumed by STAR is much the same as in scenarios in which no router fails, because all that must happen is for updates about the failed links to unreachable destinations to propagate across the network. In contrast, DSR and several other on-demand routing protocols would continue to send flood-search messages trying to reach the failed destination, which would cause a worst-case bandwidth utilization for DSR. To illustrate the impact the failure of a single destination has in DSR we have re-run the simulation scenario with 8 flows present in the network for 1800 seconds making one of the destinations fail after 900 seconds of simulated time, routers running STAR sent 1088 update packets while routers running DSR sent 3043 update packets. The existence of a single flow of data to a destination that was unreachable for 900 seconds made DSR to generate 55% more update packets while STAR generated about the same number of updates (see Table II).

IV. Conclusions

We have presented STAR, a link-state protocol that incurs less overhead than on-demand routing protocols. Because STAR can be used with any clustering mechanism proposed to date, these results clearly indicate that STAR is a very attractive approach for routing in packet-radio networks. Perhaps more importantly, the approach we have introduced in STAR for least-overhead routing opens up many research avenues, such as developing similar protocols based on distance vectors and determining how route aggregation works under LORA.

References

[1] D. Bertsekas and R. Gallager, *Data Networks*, Second Edition, Prentice-Hall, Inc., 1992.

[2] J. Broch et al, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols," *Proc. ACM MOBICOM 98*, Dallas, TX, October 1998.

[3] J.J. Garcia-Luna-Aceves and J. Behrens, "Distributed, scalable routing based on vectors of link states," *IEEE Journal on Sel. Areas in Commun.*, Vol. 13, No. 8, 1995.

[4] J.J. Garcia-Luna-Aceves et al., "Wireless Internet Gateways (WINGS)," *Proc. IEEE MILCOM'97*, Monterey, California, November 1997.

[5] J.J. Garcia-Luna-Aceves and M. Spohn, "Scalable Link-State Internet Routing," *Proc. IEEE International Conference on Network Protocols (ICNP 98)*, Austin, Texas, October 14-16, 1998.

[6] *P802.11–Unapproved Draft: Wireless LAN Medium Access Control (MAC) and Physical Specifications*, IEEE, January 1996.

[7] D. Johnson and D. Maltz, "Protocols for Adaptive Wireless and Mobile Networking," *IEEE Pers. Commun.*, Vol. 3, No. 1, February 1996.

[8] J. Jubin and J. Tornow, "The DARPA Packet Radio Network Protocols," *Proceedings of the IEEE*, Vol. 75, No. 1, January 1987.

[9] V.O.K. Li and R. Chang, "Proposed routing algorithms for the US Army mobile subscriber equipment (MSE) network," *Proc. IEEE MILCOM'86*, Monterey, California, October 1986.

[10] J. Moy, "OSPF Version 2," RFC 1583, Network Working Group, March 1994.

[11] V. Park and M. Corson, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks," *Proc. IEEE INFOCOM 97*, Kobe, Japan, April 1997.

[12] C. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," *Proc. ACM SIGCOMM 94*, London, UK, October 1994.

[13] C. Perkins, "Ad-Hoc On Demand Distance Vector (AODV) Routing," draft-ietf-manet-aodv-00.txt, November 1997.

[14] R. Ramanathan and M. Steenstrup, "Hierarchically-organized, Multihop Mobile Wireless Networks for Quality-of-Service Support," *ACM Mobile Networks and Applications*, Vol.3, No. 1, pp. 101-119, 1998.

[15] S. Murthy and J.J. Garcia-Luna-Aceves, "An Efficient Routing Protocol for Wireless Networks," *ACM Mobile Networks and Applications Journal*, Special issue on Routing in Mobile Communication Networks, 1996.

[16] M. Steenstrup (Ed.), *Routing in Communication Networks*, Prentice-Hall, 1995.

[17] The CMU Monarch Project, "Wireless and Mobility Extensions to ns-2 - Snapshot 1.0.0-beta," URL: http://www.monarch.cs.cmu.edu/cmu-ns.html, August 1998.